Experimenting with Model Solutions as a Support Mechanism

Nygren, Henrik

ACM
2019-09-05

Nygren , H , Leinonen , J , Pirttinen , N , Leinonen , A & Hellas , A 2019 , Experimenting with Model Solutions as a Support Mechanism . in Proceedings of the 1st UK & Ireland þÿ C o m p u t i n g E d u c a t i o n R e s e a r c h C o n f e r e n c e : U K I C E R . , 3 , A C M , U K I Ireland Computing Education Research Conference , Canterbury , United Kingdom , 05/09/2019 . https://doi.org/10.1145/3351287.3351288

http://hdl.handle.net/10138/315985
https://doi.org/10.1145/3351287.3351288

unspecified
acceptedVersion

Experimenting with Model Solutions as a Support Mechanism

Henrik Nygren, Juho Leinonen, Nea Pirttinen, Antti Leinonen, Arto Hellas
University of Helsinki
firstname.lastname@helsinki.fi

ABSTRACT

We describe an experiment from an introductory programming course where we provided students an opportunity to access model solutions of programming assignments they have not yet completed. Access to model solutions was controlled with coins, which students collected by completing programming assignments. The quantity of coins was limited so that students could buy solutions to at most one tenth of the course assignments. When compared to the traditional approach where access to model solutions is limited to only after the assignment is completed or the assignment deadline has passed, students seemed to enjoy the opportunity more and collecting coins motivated some students to complete more assignments. Collected coins were mostly used close to deadlines and on more difficult assignments. Overall, the use of coins and model solutions may be a viable option to providing students additional support. Data from the use of coins and model solutions could also be used to identify students who could benefit from additional guidance.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; • **Applied computing** → **Interactive learning environments**.

KEYWORDS

model solutions, worked examples, introductory programming, course performance, gamification

ACM Reference Format:
Henrik Nygren, Juho Leinonen, Nea Pirttinen, Antti Leinonen, and Arto Hellas. 2019. Experimenting with Model Solutions as a Support Mechanism. In *UK & Ireland Computing Education Research Conference (UKICER), September 5–6, 2019, Canterbury, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3351287.3351288

1 INTRODUCTION

Research from the last decade suggests that almost one third of students who attend an introductory programming course fail the course [3, 39]. Any approach that could be used to mitigate the challenges related to learning programming would be beneficial. A range of pedagogical practices that can be used to alleviate the issue exist [36], and introductory programming courses in general have been studied extensively [23]. However, many of the pedagogical practices outlined, e.g. in [36], require plenty of effort from the course teachers as well as considerable changes to the way how courses are organized.

Introductory programming courses and computer science courses in general use model solutions in two dominant ways: model solutions are used as worked examples that students study, and as a way to provide feedback and reflection. In the first case, model solutions are given to students so that they can study the problem in combination with the solution and see how a particular problem should be solved. In the second case, where model solutions are used for feedback and reflection, model solutions are typically made available after a student has solved the problem or the deadline of the problem has passed.

In this article, we describe an experiment from an introductory programming course where we set out to explore giving access to model solutions to students dynamically, and how students use the opportunity to view model solutions. The experiment blurred the distinction between worked examples and model solutions by giving students an opportunity to view model solutions of programming assignments before completing them, offering an opportunity to study how a particular problem should be solved. The experiment reported here is a continuation of an earlier experiment, described in [26], where students were able to view model solutions without any restrictions. In this work, access to model solutions was limited and model solutions could be bought using coins that students gathered from completed course assignments. The quantity of coins was limited so that students could view the model solutions of at most one tenth of the programming assignments in the course before completion. Students could still view the model solutions after they had completed the exercise or after the deadline of the exercise had passed without spending a coin.

Given the existing failure rates in programming that could be improved, any approach that could improve the situation is worth looking into. Providing access to model solutions can be seen as an additional support mechanism, which has both its upsides and downsides – students may use model solutions for support when needed, but some may also use the opportunity to avoid studying [26]. Students' approaches to learning, including mastery and performance goals, influence how students approach their tasks [42], which could also explain some of the model solution use. Moreover, coins that could be collected in the course by completing programming assignments gamifies learning, which may motivate students [6].

This article is organized as follows. In Section 2, we outline the background relevant to our experiment, discussing model solutions and examples, achievement goals, the use of tokens and hints as well as gamification. In Section 3, we explain the context, data and research questions of our work, and in Section 4 we outline the results of the experiment. The results are discussed in Section 5, and Section 6 concludes the work and outlines future directions.

UKICER, September 5–6, 2019, Canterbury, United Kingdom

ACM ISBN 978-1-4503-7257-2/19/09…$15.00
https://doi.org/10.1145/3351287.3351288

2 BACKGROUND

2.1 Model solutions and examples

A model solution is a solution to a problem that students can compare with their own solution. Benefits of model solutions include verifying results and the thought process, and they can be used to increase reflection. If a student compares and contrasts their solution with the one provided by the teacher, they can, for example, find flaws in their thought process or more optimized ways of implementing the solution.

Examples, on the other hand, provide similar content as the problems the students are working on without being a solution to the problem. Instead, students are expected to internalize and use the examples to solve the given problems. Learning materials and examples that are too challenging may hinder students' learning progress [7], and the quality of examples influences the quality of students' solutions [10].

If students are given examples, they should be relevant to the task at hand. Studying an example and completing an assignment related to the example can lead to better performance, and working on additional assignments helps learning over viewing examples [34]. Even if students are given examples of work that they are expected to do, students' background influences how they work with the examples. Students with little knowledge on the topic or poor learning approaches may copy content verbatim, while advanced students may try to solve the problems themselves referring to examples only when stuck, wanting to check a step, or wanting to avoid a more complex task such as detailed calculation [35].

One of the possible types of examples is worked examples [33], which outline the required work needed to solve the problem. Another common type of example – used in instruction – is modeling [4, 5], where a teacher shows the process needed to reach a solution. How well these work depends on students' attention, effort, and prior knowledge. For example, intricate guidance that is beneficial for novices may be extraneous for more advanced students, leading to negative outcomes [18], and thus, guidance and examples should not be given to all but only to those who need them.

2.2 Achievement goals

As mentioned previously, students' background influences how they work with examples. Achievement goal theory (AGT), developed by Dweck [8] and later revised by Harackiewics et al. [14], characterizes mastery and performance goals of students. Students may have a variety of goals or desirable outcomes from their learning experience, such as grade, interest or self-efficacy [30]. Students with mastery goals emphasize learning and mastering the course material for personal improvement, which has a positive effect on various outcomes such as deep learning, interest and self-efficacy [29, 30]. However, mastery goals do not necessarily correlate positively with grades [29].

Students with performance goals, on the other hand, emphasize the outward appearance. Performance goals can be broadly divided into two categories: appearance and normative. Students with appearance goals strive to appear knowledgeable in general, while students with normative goals strive to outperform their peers. Appearance goals tend to negatively correlate with the course grades, while normative goals may correlate positively with grades [16]. In computer science education, studies have shown that mastery goals have a positive correlation with course grades, while normative and appearance goals have may have a negative correlation with course grades [41, 43], but context also influences the observations [42].

Students with different achievement goals also have differing views on the use of work from others (plagiarism). A survey by Koul et al. [19] found that students with performance goals can be very strict in judgment towards "rationales" for plagiarizing (e.g. heavy workload, sickness, stress, embarrassment of failure), as well as sources of plagiarism (e.g. getting answers from a friend, copying directly from a textbook or the Internet). Koul et al. reason that since education can be seen as a way to "move up the social ladder", especially performance goal oriented students have an interest in keeping the competition equal and not letting anyone gain a position of privilege through fraud or a short-cut.

2.3 Gamification, tokens and hints

Gamification refers to the use of game-like elements such as points, experience or levels, or receiving achievements or tokens for completing certain goals, in non-game contexts such as educational situations [6]. Literature review data of gamification in education by Nah et al. [25] inspects gamified courses or learning materials from various contexts. They summarize learner outcomes into a few categories, the most notable ones being increased engagement [1, 2, 9, 11, 12, 20, 27] and enjoyment [1, 9]. Less common goals or learner outcomes include for example motivation, sense of achievement or accomplishment, performance, and status.

Gamification experiments have been reported also in computer science education; for example, Haaranen et al. [13] have sought to introduce badges to a data structures course, while Leppänen et al. [22] noted that even a simple progress bar with no attached rewards or grading increases students' engagement with the content. Another way to gamify learning is to introduce tokens to the learning process – for example, Spacco et al. [31] implemented a token system for their automatic assessment system Marmoset, where students had a limited set of (automatically refreshing) tokens that could be spent on testing the students' projects. Such a token system may cause students to start their work earlier.

Giving hints to students while they progress in their assignments can help students proceed towards a correct solution, but the benefits of hints are not always clear. For example, in a randomized controlled experiment, Stephen-Martinez and Fox [32] observed no benefits in a post-test when comparing test and control groups. Hints can even affect performance negatively; for example, O'Rourke et al. [28] studied hints in an educational video game environment, and found that each of the tried hint systems impacted students' performance negatively.

One of the challenges may be the way how hints are presented to students. If students do not see the overall picture but just the next step, students may not realize the full process needed to reach the solution. Providing worked examples instead could lead to better outcomes. For example, Mostafavi et al. [24] found that students who were provided worked examples in a tutoring system advanced slightly faster on their assignments than students using the same tutoring system without worked examples – furthermore, students who were provided worked examples also showed higher retention on the tutor.

3 METHODOLOGY

3.1 Experiment context

The experiment was conducted within a seven-week 5 ECTS[1] introductory programming course at the University of Helsinki. Students in the course learn the principles of procedural and object-oriented programming using Java. The course has a total of seven programming assignment sets, each corresponding to a specific week in the course. Each set has 10 to 40 programming assignments that students are expected to complete by a weekly deadline; in the beginning of the course, students have more assignments that are smaller, and towards the end of the course, the assignments grow in size and complexity.

The course uses a blended learning approach with custom online course material similar to many online course materials with embedded assignment handouts, questionnaires, program visualizations and so on. Students work on the course assignments using their own computers or the computer labs at the university, and then submit their solutions to an automated assessment service. There are no lectures in the course. Instead, students are supported in daily walk-in labs with hands-on guidance. For further details on the pedagogy, see [37, 38].

The course has an end of course exam, which is completed on a computer. The exam consists of a set of conceptual and code-reading multiple choice questions and a set of programming questions. The grading of the course is based on the completed assignments (50%) and the end of course exam (50%). Each week corresponds to approximately 7% of the total course points, and full points from a week could be gained by completing 90% of that week's assignments. The highest mark in the course can be gained by receiving over 90% of the overall marks, and students must receive at least 50% of the exam points and the overall course points to pass the course.

3.2 Experimenting with viewing model solutions

Traditionally, students in the course have been able to view model solutions to programming assignments once they have either completed the programming assignment or once the deadline has passed. An assignment is considered completed when it passess all the associated tests and is returned to the automated assessment service used in the course. In the experiment, students gained coins from completing programming assignments, which then could be used to buy model solutions for assignments that were not yet completed. In the experiment, each student received one coin for each 16 completed assignments, which corresponds to students being able to view the solutions of approximately 10% of the course assignments.

Access to model solutions was controlled using the automated assessment service, which kept record of the assignments that students had returned and consequently also the coins that each student had gathered. Whenever a student viewed a model solution, the service stored the student's id, an identifier of the programming assignment for which the model solution was viewed, and a timestamp. The service also kept track of the available coins.

[1]European Credit Transfer System: One ECTS credit corresponds to approximately 25-30 hours of work.

In the experiment, all completed assignments counted towards the grade, including the assignments for which students viewed the model solutions before completing the assignment. The students were expected to submit a solution also to any assignments for which they viewed the model solution. They were allowed to submit the model solution as their own.

3.3 Questionnaire and assignment information

At the beginning of the experiment, students were given an online survey that they were asked to answer. The survey included demographic factors (age, gender), previous programming experience measured in hours programmed, and a goal orientation questionnaire outlined in [42]. We used the questions on Mastery, Performance (appearance) and Performance (normative). Each question was answered using a 7-item Likert scale ranging from completely disagree to completely agree.

Students also answered an online research consent service where they provided consent to the use of their course data for research purposes. In addition to the questionnaire and the separate research consent, after each assignment, students were shown a voluntary prompt asking about the difficulty of the programming assignment: *On a scale from 1 to 5, how difficult was the programming assignment? (1 = very easy, 5 = very hard).*

3.4 Research questions

Our research questions for this study are as follows: RQ1. *To what extent does students' background and goal orientations correlate with model solution use?*; RQ2. *When and to what do students use the opportunity to view model solutions?*; RQ3. *How does the use of model solutions correlate with students' course outcomes?*

For the analysis, we use information on model solution views, available coins, questionnaire answers, and course outcomes. For course outcomes, we focus on students' overall performance in the programming questions of the exam. In all of the analyses, if a student has viewed the solution to a single assignment more than once, we still consider this as a single view. Furthermore, we focus on students' viewing the model solutions before completing the programming assignments.

To answer RQ1, we study to what extent students' age, gender, previous programming experience or goal orientations (mastery; performance, appearance; performance, normative) explain the use of model solutions. To answer RQ2, we contrast the timestamps of the model solution views of the two experiments and study whether students' use of model solutions differ depending on the setup. Furthermore, focusing on assignment difficulty, we study to what types of assignments students view the model solutions. To answer RQ3, we contrast students' use of model solutions with course exam outcomes.

4 RESULTS

4.1 Descriptive statistics

147 students attended the exam, provided their background details and provided consent to the use of their data for this study (approx. 57% of the initial population). This forms the set that was used for the analysis. Out of the 147 students, 96 were male, 47 were female, and 4 chose to not disclose their gender. The median age of the

Table 1: Model solution views

	min	max	mean	median	stdev
Overall	0	11	3.7	3	3.6
Normalized	0	1	0.37	0.27	0.37

students is 26 years, while the average age of the students is 29. The majority of the students in the course are studying computer science as a minor, which partly explains the relatively high average age of students.

Out of the 147 students, 61 had no previous programming experience, 52 had programmed less than 100 hours, and 34 had programmed 100 hours or more. The course exam, which was conducted using computers and contained content similar to that which the students practice during the course, went well overall. Students received on average 84.6% of the exam points with a median of 93.3% and a standard deviation of 22.6%. There was a ceiling effect in the exam.

Descriptive statistics of model solution views are presented in Table 1, which shows both the absolute number of views and the normalized number of views. The normalized number of model solution views is normalized based on the coins that the student gathered overall, i.e. ($views/coins$).

In all of the subsequent analyses, unless otherwise noted, the analysis is performed on the normalized model solution views. This way, the possibility of accessing model solutions (i.e. availability of coins) is taken into account.

4.2 Background data and model solution usage

RQ1. To what extent does students' background and goal orientations correlate with model solution use?

First, we studied whether students' demographics and background are linked with the use of model solutions. Using Pearson's correlation with age and model solutions viewed, a weak negative correlation was identified ($r = -0.25, p = 0.002$) – that is, older students are more inclined to limit their use of model solutions. Next, using the Kolmogorov-Smirnov test, we studied whether gender influences how model solutions are viewed. No statistically significant correlation between gender and model solution usage was identified. Then, studying the self-reported hours programmed and model solution usage, a statistically significant weak negative correlation was observed using Pearson's correlation. Students with previous programming experience were less likely to use model solutions ($r = -0.20, p = 0.01$).

Finally, we analyzed the connection between achievement goals and model solutions. Using Spearman's rho, we did not identify a statistically significant correlation when comparing any of the variables (1) mastery, (2) performance appearance, and (3) performance normative controlling with the number of model solution views.

4.3 Course timeline and model solution usage

RQ2. When and to what do students use the opportunity to view model solutions?

We studied the course timeline and model solution usage in two ways. First, we analyzed when the model solutions are used,

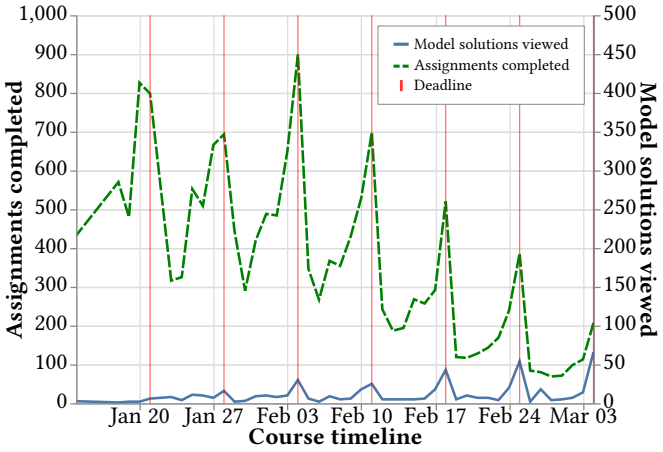

Figure 1: Completed assignments and model solution views over time. The green dashed line corresponds to completed assignments, with tick marks on the left. The blue line corresponds to viewed model solutions, tick marks on the right. The vertical red lines indicate weekly deadlines.

and then, we studied the time between a model solution view and assignment completion.

From Figure 1, we can see that students are more likely to view model solutions close to the deadline than at the beginning of each course week. Moreover, fewer model solutions are used early in the course; on the deadline day of the first course week, students complete in total approximately 800 assignments, while model solutions are viewed approximately 15 times, corresponding to using a model solution for approximately 2% of the assignments. On the other hand, on the deadline day of the final week of the course, students complete approximately 200 assignments, while model solutions are viewed approximately 60 times. This corresponds to approximately 30% of the assignments.

We looked into time between model solution views and assignment completion. Our intuition was that the time between a model solution view and assignment completion could be used to distinguish students who simply copied a model solution and students who studied the model solution. The time between model solution views and assignment completion was then correlated with exam outcomes. No statistically significant correlation was identified.

Following the analysis of the course timeline and model solution usage, we studied in what types of assignments model solutions are used. For this purpose, we used student feedback on the difficulty of the course assignments: whenever a student completes an assignment, the system asks about the difficulty of the assignment.

Using Spearman's rho to study the correlation between the average difficulty of each programming assignment and the absolute number of model solution views of each assignment, we identified a statistically significant strong correlation ($r = 0.77, p < 0.0001$). That is, the more difficult the assignment is, the more students view the model solution of the assignment before completing it. The difficulty of the programming assignment explains over 59% of the variance in the number of model solution views.

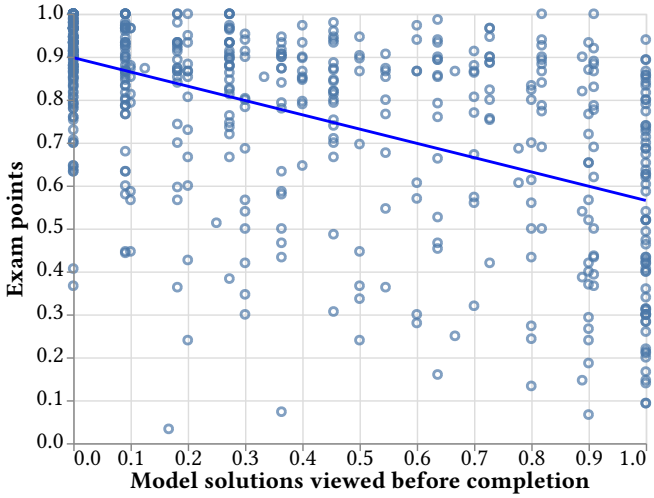

Figure 2: Exam points vs model solutions viewed. Model solution views have been normalized, accounting for the coins that students have had.

4.4 Model solution usage and course outcomes

RQ3. How does the use of model solutions correlate with students' course outcomes?

We studied the connection between model solution usage and course outcomes. Model solution use and normalized exam points are shown in Figure 2. In Figure 2, the x-axis values are normalized, meaning that the x-axis represents how many of the available coins students spent. A ratio of 0 would mean that the student did not use coins at all, while a ratio of 0.5 would mean that they spent half of their available coins.

When studying the model solution usage and course outcomes, i.e. the outcomes of the programming tasks in the exam, we identified a moderate negative correlation between model solution views and exam outcomes ($r = -0.46, p < 0.0001$). Effectively, model solution usage explains 21% of the variance in exam outcomes.

5 DISCUSSION

5.1 Background and model solutions

When studying students' background variables (age, gender, previous programming experience, goal orientations) and model solution usage, we found that (1) students with previous programming experience were more likely to use fewer model solutions, and that (2) older students are more likely to use fewer model solutions.

The observation that those with previous programming experience use fewer model solutions is in line with previous studies which suggest that students with previous programming experience may perform better in programming courses and thus may also require less help [15, 40]. The same observation was also present in the experiment where students had non-restricted access to model solutions [26]. The observation that older students are less likely to use model solutions can potentially be explained by study experience, leading them to having better study habits. On the other hand, older students may also have more exposure to programming than the younger students.

We did not identify a correlation between goal orientations and model solution usage. This was against our expectation that performance oriented students would be more likely to use model solutions than mastery oriented students as the use of model solutions can lead to more points in the course. This would have been in line with performance oriented students being more interested in showing others that they perform well. On the other hand, performance oriented students may be stricter towards plagiarism [19], which could balance the scales – we do not, however, know whether students saw using model solutions as plagiarism. Finally, the course assessment weighs the assignments and the exam evenly which likely influences how model solutions were used.

5.2 Assignment difficulty and deadlines

When analyzing course assignment difficulty and model solution usage, we found a strong correlation ($r = 0.77$) between model solution usage and assignment difficulty. Students were more likely to use model solutions in difficult assignments. This makes plenty of sense as using model solutions (a limited resource) would be wasteful for easy assignments. It is possible that some of the course assignments were too difficult for some students, which could partially explain the observation. On the other hand, assignment difficulty is linked with the time that students spend on solving the assignment [17, 21]. As such, it is possible that some students used the model solutions to get rid of laborious assignments.

When comparing the quantity of assignment submissions with model solution usage over the course, we observed that model solution usage was significantly more common close to the deadline. While students in general have a tendency to submit their work close to the deadline, the ratio of used model solutions did not correspond with the quantity of submissions over time. Using more model solutions close to the deadline could be explained by the more difficult assignments being placed at the end of each week – on the other hand, some students might simply use the model solutions to gain course points if they were running out of time. Model solution usage also increased over the course – students used more coins in the latter weeks – as with our previous experiment with unlimited access to model solutions [26]. The increased usage of model solutions towards the end of the course could be explained by students having more coins due to more completed assignments.

5.3 Exam outcomes

When looking at the correlation between the use of model solutions before completing assignments and performance in the course exam, we found a moderate negative correlation ($r = -0.46$). This means that those who viewed more model solutions before solving the assignment performed, on average, more poorly in the exam. One interpretation of this result is that our experiment where coins can be used to buy model solutions is bad for learning. We did not perform a randomized controlled trial in the course however, and as such, we do not know the actual impact of the experiment on course outcomes.

In our previous experiment reported in [26], we found a stronger negative correlation ($r = -0.71$) with the use of model solutions and exam outcomes. In the experiment reported here, students had a limited amount of model solution views, while in the previous

experiment students had non-restricted access to model solutions. It is possible that both negative correlations are explained by poor-performing students needing more help and thus having to look at more model solutions. It may be that in the experiment reported here, students who would have needed more help dropped out when they ran out of coins, whereas in the previous experiment they persisted until the exam with the help of model solutions, which could explain the difference in the magnitude of the correlations.

Another interpretation of the negative correlation between model solution usage and exam performance is that the use of coins reflects students existing study processes, providing data that could be used to identify struggling students. There exists a large body of research on the topic [15], and to our knowledge data similar to ours has not previously been used to identify struggling students. At the same time, as the use of model solutions increases over time and is at the greatest at the end of the course, using the coins as an early warning system might not be a viable option.

5.4 Student and instructor feedback

Finally, we gathered feedback on the experiment from students and instructors. Overall, the experiment was seen positive, and three main benefits were identified: 1) the ability to proceed if stuck, 2) additional incentives to complete assignments, and 3) being able to get help without asking for help.

Coins provided students the opportunity to proceed in the assignments if they got stuck. Being stuck in an assignment can be caused by a range of factors, ranging from simple syntactic issues to issues with how the problem should be approached. Using a coin provided students a peek to one way of solving the assignment, which could help realize conceptual mistakes and issues with the problem solving process.

Moreover, the gamification aspect of the experiment, i.e. students gaining coins from completed assignments, provided students additional incentives to complete more assignments. In the course, full points from each week could be gained by completing 90% of the weekly assignments, and the possibility of gathering coins encouraged students to work on additional assignments that did not count towards the grade.

Finally, we have observed that there exists a handful of students who struggle with asking for help. There are a multitude of causes for this, including shyness and being worried that asking for help somehow defines the student as a person who cannot succeed. The coins provided students an opportunity to get help, without actually asking for help.

In our previous experiment with non-restricted access to model solutions [26], we observed that students stopped coming to the walk-in labs – this may have been caused by students relying solely on the model solutions. During the experiment reported in this work, where access to model solutions was limited, the walk-in labs were again used by students.

5.5 Limitations of work

Next, we outline main limitations of this work. First, as mentioned in Section 5.3, we did not run the experiment as a randomized controlled trial with a test and control population, and as such, we can only hypothesize on causation. The results outlined in the article come from a single experiment conducted in a single course at a single institution, which means that we cannot draw conclusions on the generalizability of the results. The way how the course is organized likely affects results greatly. For example, our course had many small assignments, while there are introductory programming courses with bigger project-like assignments – such a difference could significantly influence the outcomes. Similarly, we only studied students who attended the exam and explicitly provided research consent: this, in combination with the missing randomized controlled trial, means that the study has clear selection bias. Finally, we have not explicitly looked at individual students, but limited the analysis to the population level – it is for example possible, similar to experiments with badges [13], that some students are put off by the experiment.

6 CONCLUSIONS

In this article, we described results from an experiment in an introductory programming course where students gained coins from completing programming assignments. These coins could be used to access model solutions of assignments that students had not yet completed. Our answers to the research questions are as follows:

RQ1 To what extent does students' background and goal orientations correlate with model solution use? We found that older students are somewhat less inclined to use the model solutions. Moreover, students who had programmed previously were somewhat less likely to use the model solutions. *RQ2 When and to what do students use the opportunity to view model solutions?* We found that coins were more often used close to the deadline. Furthermore, there was a strong correlation between model solution views and assignment difficulty – students were more likely to view the model solutions of difficult assignments. *RQ3 How does the use of model solutions correlate with students' course outcomes?* We found that the use of model solutions is negatively correlated with exam results. This suggests that the use of model solutions could be viewed as an indicator of students needing help, or that using model solutions may decrease learning.

Further, when discussing with students and instructors about the experiment, we identified three benefits in the use of coins: (1) coins that could be used to buy access to model solutions gave students an opportunity to proceed in the course even if they got stuck at an assignment; (2) the ability to collect coins encouraged students to complete more assignments; (3) the ability to use coins gave students who may not wish to ask for help from instructors an opportunity to get help without asking for help. This suggests that coins may lead to some students continuing in the course, even if they would have dropped out of the course otherwise.

As a part of our future work, we are working on a system where using a coin would – still – show the student a model solution, but also provide a new similar assignment. This might lead to a situation where students would get the benefits of studying the model solution, but, could not free themselves from the needed studying. Additionally, we are in the process of studying how providing students model solutions affects plagiarism – students might plagiarize less when they have access to model solutions. We are also looking into rewarding students based on unused coins, and looking for approaches to incentivize students to start working on the assignments early.

REFERENCES

[1] Adrian A. de Freitas and Michelle M. de Freitas. 2013. Classroom Live: A software-assisted gamification tool. *Computer Science Education* 23 (06 2013), 186–206.

[2] Gabriel Barata, Sandra Gama, Joaquim Jorge, and Daniel Gonçalves. 2013. Engaging Engineering Students with Gamification. *2013 5th International Conference on Games and Virtual Worlds for Serious Applications, VS-GAMES 2013*, 24–31.

[3] Jens Bennedsen and Michael E. Caspersen. 2007. Failure Rates in Introductory Programming. *SIGCSE Bull.* 39, 2 (June 2007), 32–36. https://doi.org/10.1145/1272848.1272879

[4] Allan Collins, John Seely Brown, and Ann Holum. 1991. Cognitive apprenticeship: Making thinking visible. *American educator* 15, 3 (1991), 6–11.

[5] Allan Collins, John Seely Brown, and Susan E Newman. 1989. Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. *Knowing, learning, and instruction: Essays in honor of Robert Glaser* 18 (1989), 32–42.

[6] Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. 2011. Gamification. Using Game-design Elements in Non-gaming Contexts. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2425–2428.

[7] Walter Doyle. 1983. Academic work. *Review of educational research* 53, 2 (1983), 159–199.

[8] Carol S. Dweck. 1986. Motivational Processes Affecting Learning. *American Psychologist* (1986), 1040–1048.

[9] C. A. Eleftheria, P. Charikleia, C. G. Iason, T. Athanasios, and T. Dimitrios. 2013. An innovative augmented reality educational platform using Gamification to enhance lifelong learning and cultural education. In *IISA 2013*. 1–5.

[10] Katherine Fu, Jonathan Cagan, and Kenneth Kotovsky. 2010. Design team convergence: the influence of example solution quality. *Journal of Mechanical Design* 132, 11 (2010), 111005.

[11] David Gibson, Nathaniel Ostashewski, Kim Flintoff, Sheryl Grant, and Erin Knight. 2015. Digital Badges in Education. *Education and Information Technologies* 20, 2 (2015), 403–410.

[12] Geoff Goehle. 2013. Gamification and Web-based Homework. *PRIMUS* 23 (03 2013).

[13] Lassi Haaranen, Petri Ihantola, Lasse Hakulinen, and Ari Korhonen. 2014. How (Not) to Introduce Badges to Online Exercises. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 33–38. https://doi.org/10.1145/2538862.2538921

[14] Judith Harackiewicz, Kenneth Barron, and Andrew J. Elliot. 1998. Rethinking achievement goals: When are they adaptive for college students and why? *Educational Psychologist* 33 (12 1998), 1–21.

[15] Arto Hellas, Petri Ihantola, Andrew Petersen, Vangel V Ajanovski, Mirela Gutica, Timo Hynninen, Antti Knutas, Juho Leinonen, Chris Messom, and Soohyun Nam Liao. 2018. Predicting academic performance: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 175–199.

[16] Chris Hulleman, Sheree M Schrager, Shawn M Bodmann, and Judith Harackiewicz. 2010. A Meta-Analytic Review of Achievement Goal Measures: Different Labels for the Same Constructs or Different Constructs With Similar Labels? *Psychological bulletin* 136 (05 2010), 422–49.

[17] Petri Ihantola, Juha Sorva, and Arto Vihavainen. 2014. Automatically detectable indicators of programming assignment difficulty. In *Proceedings of the 15th Annual Conference on Information technology education*. ACM, 33–38.

[18] Slava Kalyuga. 2009. The expertise reversal effect. In *Managing Cognitive Load in Adaptive Multimedia Learning*. IGI Global, 58–80.

[19] Ravinder Koul, Roy Clariana, Kalayanee Jitgarun, and Alisa Songsriwittaya. 2009. The influence of achievement goal orientation on plagiarism. *Learning and Individual Differences* 19 (12 2009), 506–512.

[20] Balraj Kumar and Parul Khurana. 2012. Gamification in Education - Learn Computer Programming with Fun. *International Journal of Computers and Distributed Systems* 2 (12 2012).

[21] Juho Leinonen, Leo Leppänen, Petri Ihantola, and Arto Hellas. 2017. Comparison of time metrics in programming. In *Proceedings of the 2017 acm conference on international computing education research*. ACM, 200–208.

[22] Leo Leppänen, Lassi Vapaakallio, and Arto Vihavainen. 2016. Illusion of Progress is Moar Addictive Than Cat Pictures. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale (L@S '16)*. ACM, New York, NY, USA, 133–136. https://doi.org/10.1145/2876034.2893388

[23] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory Programming: A Systematic Literature Review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018 Companion)*. ACM, New York, NY, USA, 55–106. https://doi.org/10.1145/3293881.3295779

[24] Behrooz Mostafavi, Guojing Zhou, Collin Lynch, Min Chi, and Tiffany Barnes. 2015. Data-driven worked examples improve retention and completion in a logic tutor. In *International Conference on Artificial Intelligence in Education*. Springer, 726–729.

[25] Fiona Nah, Qing Zeng, Venkata Rajasekhar Telaprolu, Abhishek Padmanabhuni Ayyappa, and Brenda Eschenbrenner. 2014. Gamification of Education: A Review of Literature. 401–409.

[26] Henrik Nygren, Juho Leinonen, and Arto Hellas. 2019. Non-restricted Access to Model Solutions: A Good Idea?. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 44–50.

[27] Siobhan O'Donovan, James Gain, and Patrick Marais. 2013. A Case Study in the Gamification of a University-level Games Development Course. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference (SAICSIT '13)*. ACM, 242–251.

[28] Eleanor O'Rourke, Christy Ballweber, and Zoran Popovií. 2014. Hint systems may negatively impact performance in educational games. In *Proceedings of the first ACM conference on Learning@ scale conference*. ACM, 51–60.

[29] Corwin Senko, Chris S. Hulleman, and Judith M. Harackiewicz. 2011. Achievement Goal Theory at the Crossroads: Old Controversies, Current Challenges, and New Directions. *Educational Psychologist* 46, 1 (2011), 26–47.

[30] Corwin Senko and Katie L. Tropiano. 2016. Comparing Three Models of Achievement Goals: Goal Orientations, Goal Standards, and Goal Complexes. *Journal of Educational Psychology* 108 (02 2016).

[31] Jaime Spacco, William Pugh, Nat Ayewah, and David Hovemeyer. 2006. The Marmoset project: an automated snapshot, submission, and testing system. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*. Citeseer, 669–670.

[32] Kristin Stephens-Martinez and Armando Fox. 2018. Giving hints is complicated: understanding the challenges of an automated hint system based on frequent wrong answers. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 45–50.

[33] John Sweller and Graham A Cooper. 1985. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and instruction* 2, 1 (1985), 59–89.

[34] J Gregory Trafton and Brian J Reiser. 1993. Studying examples and solving problems: Contributions to skill acquisition. In *Proceedings of the 15th conference of the Cognitive Science Society*. Citeseer, 1017–1022.

[35] Kurt VanLehn. 1998. Analogy events: How examples are used during problem solving. *Cognitive Science* 22, 3 (1998), 347–388.

[36] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. 2014. A Systematic Review of Approaches for Teaching Introductory Programming and Their Influence on Success. In *Proceedings of the Tenth Annual Conference on International Computing Education Research (ICER '14)*. ACM, New York, NY, USA, 19–26. https://doi.org/10.1145/2632320.2632349

[37] Arto Vihavainen, Matti Paksula, and Matti Luukkainen. 2011. Extreme Apprenticeship Method in Teaching Programming for Beginners. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. ACM, New York, NY, USA, 93–98. https://doi.org/10.1145/1953163.1953196

[38] Arto Vihavainen, Thomas Vikberg, Matti Luukkainen, and Jaakko Kurhila. 2013. Massive Increase in Eager TAs: Experiences from Extreme Apprenticeship-based CS1. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '13)*. ACM, New York, NY, USA, 123–128. https://doi.org/10.1145/2462476.2462508

[39] Christopher Watson and Frederick W.B. Li. 2014. Failure Rates in Introductory Programming Revisited. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (ITiCSE '14)*. ACM, New York, NY, USA, 39–44. https://doi.org/10.1145/2591708.2591749

[40] H Wood and D Wood. 1999. Help seeking, learning and contingent tutoring. *Computers & Education* 33, 2-3 (1999), 153–169.

[41] Daniel Zingaro. 2015. Examining Interest and Grades in Computer Science 1: A Study of Pedagogy and Achievement Goals. *Trans. Comput. Educ.* 15, 3 (2015), 14:1–14:18.

[42] Daniel Zingaro, Michelle Craig, Leo Porter, Brett A Becker, Yingjun Cao, Phill Conrad, Diana Cukierman, Arto Hellas, Dastyni Loksa, and Neena Thota. 2018. Achievement Goals in CS1: Replication and Extension. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 687–692.

[43] Daniel Zingaro and Leo Porter. 2016. Impact of student achievement goals on CS1 outcomes. In *Proceedings of the 47th ACM technical symposium on Computing Science Education*. ACM, 279–296.